EVENT HANDLING MECHANISM

FIELD

An embodiment of the invention relates to computer operations in general, and more specifically to an event handling mechanism.

5

BACKGROUND

In computer operations, processes generally exist to allow for special event handling. In a common example, an interrupt process may exist, in which an interrupt signal is utilized to halt normal system operations and transfer processing to the special event. However, interrupt processes by their nature utilize system require overhead and thus slow

10 down processing.

In certain environments, special event handling may instead be handled by a timer interrupt, a system that may be referred to as "interrupt free". If certain events require monitoring, this type of event handling mechanism may be modified to include periodic polling, by which a process, a connection, or other item is checked or polled at certain

15 intervals to determine whether action is required. Periodic polling may be implemented by using a periodic timer event and a function that is implemented each time the timer runs out.

However, periodic polling may impose certain costs on processor operations. Polling operations must be sufficiently frequent to perform as needed for the application.

20 However, the more frequently that polling is required for an application, the greater the resulting performance penalty.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention may be best understood by referring to the following description and accompanying drawings that are used to illustrate embodiments of the invention. In the

25 drawings:

Figure 1 illustrates a conventional interrupt process;

Figure 2 illustrates an embodiment of periodic polling in a multiprocessor system;

Figure 3 illustrates an embodiment of processor tasks in an event handling mechanism;

1

Figure 4 illustrates an embodiment of processor operations for an event handling mechanism;

Figure 5 illustrates an embodiment of processor operations for a handling mechanism utilizing another processor for event processing;

Figure 6 is a flow chart illustrating an embodiment of event handling in a computer system; and

Figure 7 is block diagram of an embodiment of an exemplary computer environment.

DETAILED DESCRIPTION

A method and apparatus are described for an event handling mechanism.

Before describing an exemplary environment in which various embodiments of the present invention may be implemented, certain terms that will be used in this application will be briefly defined:

As used herein, "bootstrap processor" means a processor to initialize a system. Initialization of a system may include booting an operating system.

As used herein, "application processor" means a processor other than a bootstrap processor to execute applications and processes in a system.

According to an embodiment of the invention, an event mechanism is provided in a multi-processor (MP) or hyper-threading (HT) computer architecture. According to embodiment of the invention, an event mechanism is implemented by allowing one physical or logical processor to engage in normal processing while directing special event operations to another physical or logical processor. In one example, one physical or logical processor may be a bootstrap processor and another physical or logical processor may be an application processor.

Under an embodiment of the invention, a computer architecture utilizes a timer interrupt for event notification. Under one embodiment, periodic polling is implemented using the timer interrupt. The periodic polling operation comprises the timer interrupt being set at certain time intervals, with a function or other action being called upon the timer reaching a time interval. According to one embodiment, one physical or logical processor executes normal processing in parallel at least in part with another physical or logical processor executing the periodic polling task.

2

Under a particular embodiment of the invention, a multi-processor (MP) or hyper-threading (HT) architecture is used to implement parallel event handling mechanism for the EFI (Extensible Firmware Interface) of Intel Corporation. (The EFI interface is described more fully in the specification, Extensible Firmware Interface Specification,

5 version 1.10, December 1, 2002, updated November 26, 2003, Intel Corporation.) The event handling mechanism utilizes an asynchronous event notification in an "interrupt free" environment, which indicates that the only interrupt available is a timer interrupt. The timer interrupt is the only asynchronous event source in the EFI environment and is made available to implement asynchronous event notification.

10 Under an embodiment of the invention, an event handling mechanism is built upon a timer interrupt. To implement the event handling mechanism, periodic polling is implemented by utilizing a periodic timer event. By calling event and timer services, program drivers will create a periodic timer event and register a callback notification function with the event. The callback function may be executed periodically (for example,

15 once every 1 ms) by the event handling mechanism. In one example, a USB (universal serial bus) driver may register a callback function that polls a USB port or ports periodically to determine whether any insertion or removal events of USB devices have occurred. However, an embodiment of an event handling mechanism may be implemented for any event that requires periodic polling.

20 While an "interrupt free" approach may simplify a computer architecture, such as the · EFI architecture, and as a result enhance system stability, there are difficulties in situations such as, for example, a network stack or other network process for which real time performance is critical. In an event handling mechanism for a network stack, a polling function may be implemented to poll the network interface card (NIC) periodically to

25 obtain incoming data in a timely manner. Because the frequency of incoming data and the length of the data that is received are generally unpredictable and because in general it is not possible to make any assumptions regarding the current condition of the network, a relative short timer period may be needed to poll with enough frequency to prevent loss of data.

30 However, a periodic polling mechanism that is implemented by leveraging a timer interrupt generally will interrupt the normal executing thread for a processor and thus result in a negative impact on the performance of the normal executing thread. The shorter the timer period utilized in periodic polling, the more of the CPU (central

3

processing unit) cycle is spent on polling and the less of the CPU cycle is available for other tasks.   In the handling of complex processes, such as TCP/IP (transmission control protocol over Internet protocol), the handler of a periodic timer event itself might become complex.   As a result, overall system performance may be greatly slowed.   At some

5   point, the loss in speed may become intolerable in, for example, a modular computing environment where there is a large amount of data transfer between computing nodes. Therefore, as periodic polling occurs more frequently, the performance penalty to the normal executing thread may increase, but the frequency of the periodic polling needs to be high enough to achieve the real time performance required by the application.

10   Under an embodiment of the invention, conflict between performance of the execution thread and performance of the event mechanism is mitigated by taking advantage of additional processing power provided in MP/HT platforms.   Under an embodiment of the invention, the event handling mechanism is shifted from the bootstrap processor (BSP) to an application processor (AP).   In this embodiment, the burden of

15   event handling is moved from bootstrap processor to application processor, freeing the operation of the bootstrap processor.

           In general for a MP/HT system, the external interrupts can be distributed to either the bootstrap processor or an application processor, the processors being equivalent for purposes of interrupt distribution.   Under an embodiment of the invention, at the

20   beginning of the platform initialization, an application processor is dedicated to handle a timer interrupt, and the timer interrupt is specified to be distributed only to that application processor.   In this embodiment, the specified application processor thus is dedicated to event handling because the event handling mechanism is built on the timer interrupt.   The application processor is responsible for executing the periodic polling task implemented by

25   exploiting a periodic timer event.   The bootstrap processor then is responsible for executing the normal computing tasks.   The normal computing tasks and the periodic polling tasks are executed in parallel, overlapping in time, at least in part.   In this embodiment, the periodic polling task doesn't result in a significant performance penalty for the normal task and the periodic polling task may be executed as frequently as needed

30   so as to meet the requirements of the particular application.

           In a conventional BIOS (basic input-output system), a bootstrap processor may be responsible for both the execution of normal tasks and event handling. The event handling, especially the periodic polling task implemented by utilizing a periodic timer event, has a

4

significant negative impact on the performance of the normal task. Under an embodiment of an event handling mechanism, the bootstrap processor only executes the normal task and a specified application processor provides for the event handling. With an event mechanism utilizing another processor for event handling, the performance of a normal

5   task and the real time performance of event handling may both be improved. In an embodiment of the invention utilizing a typical BIOS PXE (pre-boot execution environment) boot, the employment of event handling with multiple processors may both reduce boot time and improve the performance for the network stack, which may in turn reduce the loss of data packets. Under such an embodiment of the invention, there is little

10   impact to existing architecture, thus making it relatively simple to implement and deploy.

In one possible example, EFI is a single-threaded and "interrupt free" environment, with the executing thread only being interrupted by a timer interrupt. A periodic polling mechanism may be implemented in such system by leveraging the timer interrupt. A periodic polling operation would normally interrupt the normal executing thread and have

15   a negative impact on the performance of the normal executing thread. As periodic polling occurs more frequently, the performance penalty for the normal executing thread increases. However, in practice, the frequency of the periodic polling has to be high enough in order to achieve needed real time performance. In a system having multiple processors, this dilemma may be resolved by taking advantage of computing resources other than bootstrap

20   processor. Under an embodiment of the invention, the polling task is shifted to an application processor. At the beginning of platform initialization, an application processor is designated to handle the timer interrupt and the timer interrupt is specified to be distributed only to that application processor. The normal task and the polling task may be executed in parallel at least in part, the tasks overlapping in time.

25   **Figure 1** illustrates a conventional interrupt process. In Figure 1, a conventional multi-processor system may have two or more physical processors, or may have a physical processor with hyper-threading technology providing two or more logical processors. In this illustration, a first processor serves as an application processor (AP) **110** and a second processor serves as a bootstrap processor (BSP) **105**. (The processors are illustrated as

30   separate physical processors connected to a bus **115**, but a similar process may be implemented with logical processors.) External interrupts **120** can be distributed to either processor. The bootstrap processor **105** and the application processor **110** may be treated as equivalent for the purposes of interrupt distribution.

Figure 2 illustrates an embodiment of periodic polling in a multiprocessor system. In this illustration, a first physical or logical processor acts as an application processor **210** and a second physical or logical processor acts as a bootstrap processor **205**. The processors are connected to a bus **215**. In Figure 2, the computer environment utilizes a

5   timer interrupt, which is shown figuratively by a timer **220**. Periodic polling is implemented by setting the timer **220** at periodic intervals to engage a callback function **225**, thereby creating a call to a processor for periodic polling **230**. In conventional systems, a function call **235** in this instance may be directed to the bootstrap processor **205**. However, the periodic calls to the bootstrap processor may impose performance costs on

10   normal processing, particularly if an application requires that the timer intervals be relatively short. Under an embodiment of the invention, the function calls **240** are instead directed to the application processor **210**, which thus are handled in parallel with the normal processing operations of the bootstrap processor. The periodic polling process using parallel processing thus allows for less interruption of normal operations and greater

15   performance.

       Figure 3 illustrates an embodiment of processor tasks in an event handling mechanism. In Figure 3, the processing tasks are shown figuratively to illustrate the operation of an embodiment of the invention. In a non-parallel operation **305** using a timer interrupt to produce periodic polling for event management, a bootstrap processor

20   **315** may receive periodic function calls as a timer engages a callback function at certain timer intervals. However, the periodic polling tasks impose a burden on system operation, with the polling task interrupting the normal operations of the bootstrap processor **315** at certain time intervals. In addition to the loss of processing time, the bootstrap processor **315** may become overly busy and potentially may lose certain data packets. At such time,

25   an application processor **320** may be idle.

       In a parallel operation **310**, the bootstrap processor **325** may not be required to process a polling task for a periodic polling function. The bootstrap processor **325** may thus be capable of concentrating on the processing of the normal task, which, as illustrated in Figure 3, may be completed more quickly. The application processor **330** may process

30   the polling task, which in this illustration is shown as alternating idle periods with polling task periods.

       Figure 4 illustrates an embodiment of processor operations for an event handling mechanism. Figure 4 specifically demonstrates a TCP/IP (transmission control protocol

over Internet protocol) network stack implemented using an event handling mechanism. In Figure 4, a bootstrap processor **405** receives periodic function calls to enable a periodic polling function. An application processor **410** is present, but is idle at this time. Also illustrated is shared memory **415** that may be used to transfer data between the bootstrap

5   processor **405** and the application processor **410**, although other methods of data transfer may also be utilized. A polling loop **420** is used by an application to periodically poll the network stack. In this particular illustration, a polling loop **420** may initiate a process with a network protocol layer **425** and a network interface layer **430**. Upon a determination that there is data to be transferred from the network stack, there are

10   communications back between the network interface layer **430** and the network protocol layer **425**. A data packet **432** may be transferred for processing by the bootstrap processor **405**, which may require certain actions such as writing to disk **435**, executing commands **440**, post-processing operations **450**, such as operations with regard to an FTP (file transfer protocol) server, and other functions **445**. However, the workload placed on

15   the bootstrap processor **405** may be significant, particularly if the polling loop **420** runs at a short interval. In certain cases, the periodic polling tasks may create situations in which data packets are lost or performance is otherwise unacceptable.

Figure 5 illustrates an embodiment of processor operations for an event handling mechanism utilizing another processor for event processing. Figure 5 demonstrates a

20   TCP/IP network stack implemented using a parallel event handling mechanism. In this embodiment, an application processor **510** receives periodic function calls to enable a periodic polling function while a bootstrap processor **505** is engaged in normal processing. Also illustrated is shared memory **515** that may be used to transfer data between the bootstrap processor **505** and the application processor **510**, although other methods of data

25   transfer may be utilized. A polling loop **520** periodically polls the network interface. In this particular illustration, the polling loop **520** provides for communications between a network protocol layer **525** and a network interface layer **530**. Upon determination that there is data to be transferred from the network stack, there are communications back between the network interface layer **530** and the network protocol layer **525**. A data

30   packet **535** may be transferred from the application processor **510** to the bootstrap processor **505** for processing, which may engage in actions such as writing to disk **540**, executing commands **545**, post-processing operations **555**, such as operations with regard to an FTP (file transfer protocol) server, and other operations **550**. Under an embodiment

7

of the invention, the workload placed on the bootstrap processor **505** may be significantly reduced in comparison with an event handling mechanism that relies only on a bootstrap processor for the performance of a polling loop **520**. With the parallel event handling mechanism, there is more opportunity for the conduct of unrelated work **560** by the

5   bootstrap processor **505**, thereby improving performance and reducing the likelihood of data loss.

Figure 6 is a flow chart illustrating an embodiment of parallel event handling in a computer system. Figure 6 illustrates certain processes for a bootstrap processor and an application processor. Figure 6 contains a simplified process to illustrate an embodiment

10   of parallel event handling and is not intended to illustrate all operations of the processors. In this illustration, the bootstrap processor **605** provides for normal processing **610**. If a special event related to the periodic polling requires processing **615**, such as a data packet from a network stack, the event data is processed **620**. If there is no special event, the normal processing **610** will continue.

15   For an application processor **630**, a polling function may be loaded **635**, the polling function to provide for periodic polling. A timer interrupt is set for certain time intervals **640**. The lengths of the time intervals are dependent upon the needs of the application in question. (In a particular embodiment, the timer intervals may be of equal length, such as repeating every *n* microseconds, but embodiments of the invention are not limited to equal

20   time intervals.) After some time delay **645**, there is a determination whether the timer has run out **650**, indicating that the timer has run for a specified interval and the timer interrupt occurs. If not, the delay continues. If so, the polling function operation is performed **655**. If no action is required **660**, the time delay for the next time interval runs **645**. If there is a positive result indicating that action is required **660**, such as a data packet that

25   requires processing, there is a communication to the bootstrap processor **665**, thereby allowing the processing of the special event **620**.

Embodiments of the invention may be implemented in various computer environments. **Figure 7** is block diagram of an embodiment of an exemplary computer environment. Under an embodiment of the invention, a computer **700** comprises a bus

30   **705** or other communication means for communicating information, and a processing means such as one or more physical processors **710** (shown as **711**, **712** and continuing through **713**) coupled with the bus **705** for processing information. Each of the physical processors may include multiple logical processors. The physical or logical processors

8

may operate in parallel, such as in special event handling being performed in parallel with normal thread execution. Each processor may include an execution unit and logic for the operation of certain functions.

The computer **700** further comprises a random access memory (RAM) or other dynamic storage device as a main memory **715** for storing information and instructions to be executed by the processors **710**. Main memory **715** also may be used for storing temporary variables or other intermediate information during execution of instructions by the processors **710**. The computer **700** also may comprise a read only memory (ROM) **720** and/or other static storage device for storing static information and instructions for the processors **710**.

A data storage device **725** may also be coupled to the bus **705** of the computer **700** for storing information and instructions. The data storage device **725** may include a magnetic disk or optical disc and its corresponding drive, flash memory or other nonvolatile memory, or other memory device. Such elements may be combined together or may be separate components, and utilize parts of other elements of the computer **700**.

The computer **700** may also be coupled via the bus **705** to a display device **730**, such as a liquid crystal display (LCD) or other display technology, for displaying information to an end user. In some environments, the display device may be a touch-screen that is also utilized as at least a part of an input device. In some environments, display device **730** may be or may include an auditory device, such as a speaker for providing auditory information. An input device **740** may be coupled to the bus **705** for communicating information and/or command selections to the processors **710**. In various implementations, input device **740** may be a keyboard, a keypad, a touch-screen and stylus, a voice-activated system, or other input device, or combinations of such devices. Another type of user input device that may be included is a cursor control device **745**, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processors **710** and for controlling cursor movement on display device **730**.

A communication device **750** may also be coupled to the bus **705**. Depending upon the particular implementation, the communication device **750** may include a transceiver, a wireless modem, a network interface card, or other interface device. The computer **700** may be linked to a network or to other devices using the communication device **750**, which may include links to the Internet, a local area network, or another environment.

9

In the description provided above, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that the present invention may be practiced without some of these specific details. In other instances, well-known structures

5   and devices are shown in simplified block diagram form.

The present invention may include various processes.   The processes of the present invention may be performed by hardware components or may be embodied in machine-executable instructions, which may be used to cause a general-purpose or special-purpose processor or logic circuits programmed with the instructions to perform

10   the processes. Alternatively, the processes may be performed by a combination of hardware and software.

Portions of the present invention may be provided as a computer program product, which may include a machine-readable medium having stored thereon instructions, which may be used to program a computer (or other electronic devices) to perform a process

15   according to the present invention.   The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs (compact disk read-only memory), and magneto-optical disks, ROMs (read-only memory), RAMs (random access memory), EPROMs (erasable programmable read-only memory), EEPROMs (electrically-erasable programmable read-only memory), magnet or optical cards, flash memory, or other type of

20   media/machine-readable medium suitable for storing electronic instructions.   Moreover, the present invention may also be downloaded as a computer program product, wherein the program may be transferred from a remote computer to a requesting computer by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection).

25   Many of the methods are described here in their most basic form, but processes can be added to or deleted from any of the methods and information can be added or subtracted from any of the described messages without departing from the basic scope of the present invention.   It will be apparent to those skilled in the art that many further modifications and adaptations can be made.   The particular embodiments are not provided to limit the

30   invention but to illustrate it.   The scope of the present invention is not to be determined by the specific examples provided above but only by the claims below.

It should also be appreciated that reference throughout this specification to "one embodiment" or "an embodiment" means that a particular feature may be included in the

practice of the invention. Similarly, it should be appreciated that in the foregoing description of exemplary embodiments of the invention, various features of the invention are sometimes grouped together in a single embodiment, figure, or description thereof for the purpose of streamlining the disclosure and aiding in the understanding of one or more

5   of the various inventive aspects. This method of disclosure, however, is not to be interpreted as reflecting an intention that the claimed invention requires more features than are expressly recited in each claim. Rather, as the following claims reflect, inventive aspects lie in less than all features of a single foregoing disclosed embodiment. Thus, the claims are hereby expressly incorporated into this description, with each claim standing on

10   its own as a separate embodiment of this invention.